



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/634,937	08/05/2003	Dale Koetke	MS#303250 . 02 (5216 . 1)	7916
38779 7590 06/18/2007 SENNIGER POWERS (MSFT) ONE METROPOLITAN SQUARE, 16TH FLOOR ST. LOUIS, MO 63102			EXAMINER WEINTROP, ADAM S	
			ART UNIT	PAPER NUMBER
			2145	
			NOTIFICATION DATE	DELIVERY MODE
			06/18/2007	ELECTRONIC

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

uspatents@senniger.com

## Office Action Summary

Application No.

10/634,937

Applicant(s)

KOETKE ET AL.

Examiner

Adam S. Weintrop

Art Unit

2109

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

### Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

### Status

- 1) ☒ Responsive to communication(s) filed on 05 August 2003.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

### Disposition of Claims

- 4) ☒ Claim(s) 1-50 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-50 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

### Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 05 August 2003 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

### Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
  - ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
  - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

### Attachment(s)

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☒ Information Disclosure Statement(s) (PTO/SB/08)  
Paper No(s)/Mail Date 6/8/06, 8/9/06.
- 4) ☐ Interview Summary (PTO-413)  
Paper No(s)/Mail Date. \_\_\_\_\_
- 5) ☐ Notice of Informal Patent Application
- 6) ☐ Other: \_\_\_\_\_

## DETAILED ACTION

### *Claim Objections*

1. **Claims 1-10 and 34-37** are objected to because of the following informalities:

Regarding **claim 1**, the term "a client" on claim line 4 has already been defined and should be replaced with --the client-- to improve the clarity of the claim language.

Regarding **claims 4-6**, the terms "latency information" on claim lines 2 have already been defined and should be replaced with --the latency information-- to improve the clarity of the claim language.

Regarding **claim 34**, the terms "client system" on claim lines 5-7 should be replaced with --computerized client system-- to improve the clarity of the claim language.

Appropriate correction is required.

### *Claim Rejections - 35 USC § 112*

2. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

3. **Claims 8, 22, 23, and 24** are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Regarding **claims 8, 22, 23, and 24**, the claim ends with the phrase "and subcombinations thereof". This phrase creates confusion as to what the applicant is trying to claim as the invention because the specific proportions of the subcombination are not clearly defined.

***Claim Rejections - 35 USC § 101***

4. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

**Claims 10, 27, and 42-50** are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Regarding **claims 10, 27, and 42-50**, the claims are drawn towards a computer-readable medium. The computer readable medium can be viewed as a carrier wave. In order for a claim to be statutory, it must fall into one of the statutory categories of invention as an apparatus, manufacture, method, or composition of matter. Electromagnetic signals do not fall into one of these categories, and therefore are non-statutory subject matter.

Regarding **claims 42-50**, the claims are drawn towards a data structure. This data structure performs no data manipulation, and is therefore seen as non-functional descriptive material. The claims are drawn to several fields that are only placeholders for data to be entered, but no output happens. In order for a claim to be statutory, it must fall into one of the statutory categories of invention as an apparatus, manufacture,

method, or composition of matter. Non-functional descriptive material does not fall into one of these categories and is therefore non-statutory subject matter.

***Claim Rejections - 35 USC § 102***

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

6. **Claims 1-2, 4-6, 10, 12-16, 21, 27, 30-32, and 38** are rejected under 35 U.S.C. 102(b) as being anticipated by Bland et al. (US 5,732,218).

Regarding **claim 1**, Bland et al. anticipates:

A method for reporting latency information as perceived by a client in a client server system (Abstract), the method comprising:

dispatching a first request from a client to a server (column 5, lines 45-48, with a client sending a request to a server);

receiving a first response from the server, wherein the first response corresponds to the first request (column 5, lines 50-51, with the response being the page requested);

measuring a latency from the client's dispatch of the first request to the client's receipt of the first response from the server (column 4, lines 11-12, with the duration of delay seen as a latency measurement from a request to a response);

appending the latency information to a second request (column 5, lines 56-62, with the sending of the data to a URL seen as appending the data to a request);

Art Unit: 2109

dispatching the second request with the latency information from the client to the server (column 5, lines 56-62, with the sending of the data to a URL seen as dispatching the request).

Regarding **claim 2**, Bland et al. anticipates:

The method of claim 1, wherein the client is a messaging client, and wherein the server is a messaging server (Figure 2, where clients and servers interact, and this is seen as messaging back and forth, or messaging clients and servers).

Regarding **claim 4**, Bland et al. anticipates:

The method of claim 1, further comprising accumulating latency information from the client at the server (column 4, lines 11-12, where latency information is computed, and column 4, lines 60-64, where the client distributes the data to a server).

Regarding **claim 5**, Bland et al. anticipates:

The method of claim 1, further comprising accumulating latency information for a plurality of clients at the server (column 4, lines 11-12, where latency information is computed, and column 4, lines 60-64, where the client distributes the data to a server, and column 6, lines 2-4, where a plurality of clients gather data).

Regarding **claim 6**, Bland et al. anticipates:

Art Unit: 2109

The method of claim 1, further comprising accumulating latency information for a plurality of clients and a plurality of servers at the server (column 4, lines 11-12, where latency information is computed, and column 4, lines 60-64, where the client distributes the data to a server, and column 6, lines 2-4, where a plurality of clients and servers gather data to send to the service manager).

Regarding **claim 10**, Bland et al. anticipates:

A software program embodied on a computer readable medium, wherein the software program is executable to perform the method of claim 1 (column 2, line 66-column 3, line 6, where clients and servers execute software to perform the method of claim 1).

Regarding **claim 12**, Bland et al. anticipates:

A computer-implemented method, comprising: sending a first request from a client to a server (column 5, lines 45-48, with a client sending a request to a server); recording, at the client, a request initiation time for the first request (column 4, lines 11-20, where latency information is monitored by subtracting request time and response time, and this inherently must record a request initiation time); receiving, at the client, a first response from the server corresponding to the first request (column 5, lines 50-51, with the response being the page requested); recording, at the client, a response received time for the first response (column 4, lines 11-20, where latency information is monitored by subtracting request time and response time, and this inherently must record a response time);

calculating a round trip latency for the first request/response pair comprising a difference between the response received time for the first response and the request initiation time for the first request (column 4, lines 11-20, where latency information is monitored by subtracting request time and response time); and sending a second request from the client to the server, the second request comprising performance data, and the performance data comprising the round trip latency for the first request/response pair (column 5, lines 56-62, with the sending of the data to a URL seen as sending the performance data with the latency information to the server).

Regarding **claim 13**, Bland et al. anticipates:

The method according to claim 12, further comprising:

receiving the first request at the server (column 3, lines 47-49, where client requests are received on a server);

recording, at the server, a request received time for the first request (column 3, lines 47-49, where the duration of delay is computed, and inherently this must record the request received time);

recording, at the server, a response initiation time for the first response (column 3, lines 47-49, where the duration of delay is computed, and inherently this must record the response initiation time);

calculating a server processing time for the first request/response pair comprising a difference between the response initiation time for the first response and the request received time for the first request (column 3, lines 47-49, where the duration of delay is



computed from client requests and server response time) ;  
sending the first response to the client (column 5, lines 50-51, with the response being the page requested);  
sending the server processing time for the first request/response pair to the client (column 4, lines 11-20, where a request/response pair is measured for timing and that data is measured at the client, seen as sending the server processing time back to the client); and  
recording, at the client, the server processing time for the first request/response pair (column 4, lines 11-20, where the server processing time is computed, seen as recording it).

Regarding **claim 14**, Bland et al. anticipates:

The method according to claim 13, wherein the first response comprises the server processing time for the first request/response pair (column 4, lines 11-20, where a request/response pair is measured for timing and that data is measured at the client, seen as sending the server processing time back to the client in a first response)

Regarding **claim 15**, Bland et al. anticipates:

The method according to claim 14, wherein the performance data further comprises the server processing time for the first request/response pair (column 5, lines 56-62, with the sending of the data to a URL seen as sending performance data as another request to the server).

Regarding **claim 16**, Bland et al. anticipates:

The method according to claim 15, further comprising:

receiving the second request at the server (column 6, lines 2-4, where the data is received at the server);

parsing the performance data from the second request (column 6, lines 2-4, where generating reports is seen as parsing the data);

and

updating at least one computer system memory resident performance data accumulator with the performance data (column 6, lines 2-4, where generating reports is also seen as updating a memory accumulator).

Regarding **claim 21**, Bland et al. anticipates:

The method according to claim 12, wherein the performance data further comprises client server communications session invariant performance data context (column 4, lines 54-59, where the performance data sent back to the server includes client demographics, seen as session invariant data context since it does not depend on the communication session in particular, but depends on the client's demographics), and the performance data context comprising at least one performance data context identifier (column 4, lines 54-59, where the context here includes type of client hardware, which is seen as an identifier).

Regarding **claim 27**, Bland et al. anticipates:

Art Unit: 2109

A computer readable medium having thereon computer executable instructions for performing the method of claim 12 (column 2, line 66-column 3, line 6, where clients and servers execute software to perform the method of claim 12).

Regarding **claim 30**, Bland et al. anticipates:

A computer-implemented method, comprising:

sending a first request from a client to a first server (column 5, lines 45-48, with a client sending a request to a server);

Recording, at the client, a request initiation time for the first request (column 4, lines 11-20, where latency information is monitored by subtracting request time and response time, and this inherently must record a request initiation time);

receiving, at the client, a first response from the first server corresponding to the first request (column 5, lines 50-51, with the response being the page requested);

recording, at the client, a response received time for the first response (column 4, lines 11-20, where latency information is monitored by subtracting request time and response time, and this inherently must record a response time);

calculating a round trip latency for the first request/response pair comprising a difference between the response received time for the first response and the request initiation time for the first request (column 4, lines 11-20, where latency information is monitored by subtracting request time and response time); and

sending a second request from the client to a second server, the second request comprising performance data, and the performance data comprising the round trip

Art Unit: 2109

latency for the first request/response pair (column 5, lines 56-62, with the sending of the data to a URL seen as sending the performance data with the latency information to the server).

Regarding **claim 31**, Bland et al. anticipates:

The method according to claim 30, further comprising:

receiving the second request at the second server (column 6, lines 2-4, where the data is received at the server);

Parsing the performance data from the second request (column 6, lines 2-4, where generating reports is seen as parsing the data);

and

updating, with the performance data, at least one computer system memory resident performance data accumulator associated with the first server (column 6, lines 2-4, where generating reports is also seen as updating a memory accumulator).

Regarding **claim 32**, Bland et al. anticipates:

A computer-implemented method, comprising:

Sending a first request from a client to a first server (column 5, lines 45-48, with a client sending a request to a server);

Recording, at the client, a request initiation time for the first request (column 4, lines 11-20, where latency information is monitored by subtracting request time and response time, and this inherently must record a request initiation time);

Art Unit: 2109

Receiving, at the client, a first response from the first server corresponding to the first request (column 5, lines 50-51, with the response being the page requested);

Recording, at the client, a response received time for the first response (column 4, lines 11-20, where latency information is monitored by subtracting request time and response time, and this inherently must record a response time);

Calculating a round trip latency for the first request/response pair comprising a difference between the response received time for the first response and the request initiation time for the first request (column 4, lines 11-20, where latency information is monitored by subtracting request time and response time);

Storing, at the client, performance data associated with the first request/response pair and a performance data storage time, the performance data comprising the round trip latency for the first request/response pair (column 4, lines 9-12, with the client monitoring performance latency data, seen as storing since the data is collected, and also column 4, lines 54-59, with client demographics collected, such as time zone, seen as performance data storage time, since it places a time on a client); and  
sending a second request from the client to a second server (column 5, lines 56-62, with the sending of the data to a URL seen as sending the performance data with the latency information to the server); and

If a difference between a request initiation time for the second request and the storage time for the performance data associated with the first request/response pair is less than a maximum performance data age threshold then incorporating the performance data associated with the first request/response pair into the second request (column 5,

lines 14-23, where the client sends the data to the server in the second request after data is in the file, seen as a storage time for the first request/response pair, and then it sends it periodically, seen as a request initiation time for the second request, thus the data is stored and sent in intervals every 30 minutes for example, and this is seen as placing a data age threshold on the data).

Regarding **claim 38**, Bland et al. anticipates:

A computerized server system, comprising: a performance data stream parse module configured to, at least, parse client-generated performance data from an incoming data stream (column 6, lines 2-4, where generating reports is seen as parsing the data, and the reports come from data sent to the server from the clients, seen as an incoming data stream);

at least one server system memory resident performance data accumulator (column 6, lines 2-4, where the report is also seen as a memory accumulator); and

a performance data report module configured to, at least, update the at least one server system memory resident performance data accumulator from performance data corresponding to the parsed client-generated performance data (column 6, lines 2-4, where generating reports is also seen as updating a memory accumulator).

### ***Claim Rejections - 35 USC § 103***

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

Art Unit: 2109

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

8. **Claims 3 and 11** are rejected under 35 U.S.C. 103(a) as being unpatentable over Bland et al. (US 5,732,218) in view of Burgess et al. (US 5,696,701).

Regarding **claim 3**, Bland et al. discloses all of the limitations as described above except for using remote procedure calls as the requests. The general concept of using remote procedure calls with monitoring a computer is well known in the art as illustrated by Burgess et al. Burgess et al. teaches that a computer can be monitored, and this performance data can be passed from one computer to another with remote procedure calls (column 9, lines 42-56, where the event queue is on the monitored computer, and that passes messages to the monitoring listener with RPCs). It would have been obvious to one of ordinary skill in the art at the time of invention to modify Bland et al. with using RPCs as the message passing structure as taught by Burgess et al. in order to facilitate tracking of the monitored data as noted in Burgess et al.'s disclosure in column 2, lines 17-20.

Regarding **claim 11**, Bland et al. teaches:

A method for determining performance in a client server system, the method comprising:

monitoring on a client a status of each of a plurality of requests sent to a server (column 5, lines 45-48, with a client sending a request to a server);  
appending information regarding the status of the request to at least one subsequent

Art Unit: 2109

request that is sent to the server (column 5, lines 56-62, with the sending of the data to a URL seen as sending the performance data with the latency information to the server); and

generating aggregate statistics on the server regarding performance of the server as perceived by the client based upon the status (column 6, lines 2-4, where generating reports is seen as aggregating statistics on the server of client side monitoring).

Bland et al. does not teach monitoring RPC calls in a monitoring system and generating alerts if the statistics exceed a threshold.

The general concept of using remote procedure calls with monitoring a computer and generating alerts if a threshold is exceeded is well known in the art as illustrated by Burgess et al. Burgess et al. teaches that a computer can be monitored, and this performance data can be passed from one computer to another with remote procedure calls (column 9, lines 42-56, where the event queue is on the monitored computer, and that passes messages to the monitoring listener with RPCs). Burgess et al. also teaches alerting the system if a monitoring performance value is exceeded (column 6, lines 40-49, where an alertable level is seen as a threshold). It would have been obvious to one of ordinary skill in the art at the time of invention to modify Bland et al. with using RPCs as the message passing structure and alerting on exceeded thresholds as taught by Burgess et al. in order to facilitate tracking of the monitored data as noted in Burgess et al.'s disclosure in column 2, lines 17-20.



9. **Claims 7 and 9** are rejected under 35 U.S.C. 103(a) as being unpatentable over Bland et al. (US 5,732,218) in view of Cote et al. (US 5,870,556).

Regarding **claims 7 and 9**, Bland et al. discloses all of the limitations as described above except for having the servers be a different type of messaging server as required by claim 7 or having the client be an email client and the server be an email server as required by claim 9. The general concept of using email clients and servers and different types of messaging servers in a monitoring system is well known in the art as illustrated by Cote et al. Cote et al. teaches that a monitoring system can be used with a messaging server and client (column 1, lines 38-40, with monitoring, and column 1, lines 4-7, with messaging clients and servers). Cote et al. also teaches that the monitoring system can be used with multiple types of servers (column 1, lines 62-64). It would have been obvious to one of ordinary skill in the art at the time of invention to modify Bland et al. with monitoring messaging servers with messaging clients and using different types of messaging servers as taught by Cote et al. in order to reduce failure associated with messaging systems as noted in Cote et al.'s disclosure in column 1, lines 23-40.

10. **Claim 8** is rejected under 35 U.S.C. 103(a) as being unpatentable over Bland et al. (US 5,732,218) and Cote et al. (US 5,870,556) as applied to claim 7 above, and further in view of Chan et al. (US 7,016,909).

Regarding **claim 8**, Bland et al. and Cote et al. teach all of the limitations as described above except for having the plurality of servers include mail servers, public folder servers, and calendar/scheduling servers. The general concept of including these functions is well known in the art as illustrated by Chan et al. Chan et al. teaches that an Exchange server can be used and accessed by a client and it includes the functionality of email, calendar, and public folders (column 2, lines 58-63). It would have been obvious to one of ordinary skill in the art at the time of invention to modify Bland et al. and Cote et al. with using Exchange server as taught by Chan et al. in order to test all kinds of messaging servers.

11. **Claims 28 and 29** are rejected under 35 U.S.C. 103(a) as being unpatentable over Cote et al. (US 5,870,556) in view of Burgess et al. (US 5,696,701).

Regarding **claim 28**, Cote et al. teaches:

A computer-implemented method, comprising:

Sending a first request from a client to a server (column 6, lines 1-11, where a client sends a request to a server);

Recording, at the client, an error condition corresponding to the first request (column 4, lines 15-22, with the client software reporting a deficiency if the server's round trip time is excessive, seen as an error condition corresponding to the request).

Cote et al. does not teach sending a second request from the client to the server, with the second request comprising: an indication of at least one service desired of the server by the client, and performance data, and the performance data comprising the

Art Unit: 2109

error condition corresponding to the first request. The general concept of sending a second request with a service request and including error conditions of the first request is well known in the art as illustrated by Burgess et al. Burgess et al. teaches that a monitoring computer can generate events if a monitored computer's performance has reached a critical level (column 4, lines 4-15). This event is sent to the listener computer, which is seen as a second request, since the monitored computer has already been monitored and now it is reporting its findings to the listener, also seen as the request comprising an indication of a service desired (column 9, lines 26-41, where once performance data is available, the computer make a RPC to a gather the performance data). The request will also include the error conditions (column 11, lines 5-8, where the alerts are RPCs sent). It would have been obvious to one of ordinary skill in the art at the time of invention to modify Cote et al. with using second requests filled with the error information and an indication of a service requested as taught by Burgess et al. in order to facilitate tracking of the monitored data as noted in Burgess et al.'s disclosure in column 2, lines 17-20.

Regarding **claim 29**, Cote et al. and Burgess et al. teach all of the limitations as described above, with Burgess et al. further teaching: receiving the second request at the server (column 11, lines 28-42, where the statistics gathering thread gathering the performance data by RPCs and this is seen as receiving the second request at the server since the RPCs are returned with the performance data); parsing the performance data from the second request (column 11, lines 62-66, where the ASCII

Art Unit: 2109

agent takes the data obtained, seen as parsing the data obtained from the request); classifying the first request based on the error condition corresponding the first request (column 12, lines 9-12, where data is processed from the event queue and some data is forwarded further, seen as classifying based on the error condition, since errors can be placed in the event queue as seen in column 4, lines 4-15); and updating at least one computer system memory resident performance data accumulator associated with the request class (column 12, lines 1-5, with the SQL database storing the data from the event queue, seen as updating a memory resident performance data accumulator). It would have been obvious to one of ordinary skill in the art at the time of invention to modify Cote et al. with parsing the second request, classifying and parsing, and then updating a database as taught by Burgess et al. in order to facilitate tracking of the monitored data as noted in Burgess et al.'s disclosure in column 2, lines 17-20.

12. **Claims 17-20 and 41** are rejected under 35 U.S.C. 103(a) as being unpatentable over Bland et al. (US 5,732,218) in view of Chong et al. (US 2004/0064552).

Regarding **claim 17**, Bland et al. discloses all of the limitations as described above and Bland et al. further teaches having the performance data accumulator comprise a request count accumulator (column 3, lines 53-54), a request class accumulator (column 3, lines 61-63, with browser type seen as a class), and a request rate accumulator (column 3, lines 44-46, with busyness seen as request rate). Bland et al. does not teach having the performance data accumulator comprise a request class rate accumulator and a count of a performance data parameter value exceeding a threshold accumulator. The general concept of including these accumulators in a

performance data accumulator is well known in the art as illustrated by Chong et al. Chong et al. teaches a monitoring system where a count is increased if a request is of a certain type (section 0055, lines 8-16), and also a count is increased if a threshold is exceeded (section 0057, lines 3-6). It would have been obvious to one of ordinary skill in the art at the time of invention to modify Bland et al. with using these different accumulator fields as taught by Chong et al. in order to improve application performance by monitoring all type of configurations as noted in Chong et al.'s disclosure in section 0004, lines 9-13.

Regarding **claim 18**, Bland et al. and Chong et al. teach all of the limitations as described above and Chong et al. further teaches generating an event log entry if the performance data triggers a performance data event and the performance data event would not contribute to a performance data event storm (section 0057, lines 1-10, where logging occurs after thresholds are passed, and the event storm has not happened as seen in section 0057, lines 15-17, where a threshold has to be passed a certain number of times in order to generate an escalation of alerts). It would have been obvious to one of ordinary skill in the art at the time of invention to modify Bland et al. and Chong et al. with the further teachings of Chong et al. in order to improve application performance by monitoring all type of configurations as noted in Chong et al.'s disclosure in section 0004, lines 9-13.

Regarding **claim 19**, Bland et al. and Chong et al. teach all of the limitations as described above and Chong et al. further teaches wherein triggering a performance data event comprises a ratio of successful requests to total requests dropping below a minimum and an average request latency rising above a maximum (section 0062, where logging occurs based on the metrics in the table, where one metric is Faults and one is throughput, and by using these two, you can obtain the ratio of successful request to total requests, and another metric is response time, which is seen as latency). Thresholds can be set to any of these metrics (section 0056, lines 9-15). It would have been obvious to one of ordinary skill in the art at the time of invention to modify Bland et al. and Chong et al. with the further teachings of Chong et al. in order to improve application performance by monitoring all type of configurations as noted in Chong et al.'s disclosure in section 0004, lines 9-13.

Regarding **claim 20**, Bland et al. and Chong et al. teach all of the limitations as described above and Chong et al. further teaches wherein contributing to a performance data event storm comprises generating more than a maximum number of performance data events in a particular performance data event class during a period of time (section 0057, lines 1-10, where logging occurs after thresholds are passed, and the event storm happens after a maximum number of threshold occurrences have occurred as seen in section 0057, lines 15-17, where a threshold has to be passed a certain number of times in order to generate an escalation of alerts, seen as a storm). It would have been obvious to one of ordinary skill in the art at the time of invention to modify Bland et al.

and Chong et al. with the further teachings of Chong et al. in order to improve application performance by monitoring all type of configurations as noted in Chong et al.'s disclosure in section 0004, lines 9-13.

Regarding **claim 41**, Bland et al. teaches all of the limitations as described above except for a performance data event log; a performance data event trigger database comprising at least one criteria set corresponding to criteria for triggering a performance data event; and wherein the performance data report module is further configured to, at least: for each criteria set in the performance data event trigger database, determine if the criteria set is met as a consequence of the parsed client-generated performance data; and for each criteria set that is met: generate a performance data event; and enter the performance data event in the performance data event log. The general concept of including an event log and a trigger database is well known in the art as illustrated by Chong et al. Chong et al. teaches a monitoring system that has thresholds set. Chong et al. teaches a performance data event log (section 0057, lines 8-10) and a performance data event trigger database (section 0056, lines 6-15, with thresholds for performance data seen as a performance data event trigger database with set criteria). Chong et al. also teaches a report module that tests each criteria for a performance data trigger and then generate a performance data event and log the event (section 0057, lines 1-15, where if a threshold is met, an event is generated and it is logged. It would have been obvious to one of ordinary skill in the art at the time of invention to modify Bland et al. with the teachings of Chong et al. in order to improve application

performance by monitoring all type of configurations as noted in Chong et al.'s disclosure in section 0004, lines 9-13.

13. **Claim 22** is rejected under 35 U.S.C. 103(a) as being unpatentable over Bland et al. (US 5,732,218) in view of "Network Diagnostics Tools Feature Overview".

Regarding **claim 22**, Bland et al. discloses all of the limitations as described above including having the context include client computer system host name (column 4, lines 5-8). Bland et al. does not teach having the context comprise a client user name, network adaptor, adaptor speed, and protocol address. The general concept of including these configuration details with the contexts is well known in the art as illustrated by "Network Diagnostics Tools Feature Overview". "Network Diagnostics Tools Feature Overview" teaches that certain configuration details can be included in tests run to monitor a network. The user name is shown on page 10, under Computer Information as "CSName". The network adaptor name and speed is shown on page 20 in Figure 9. The network protocol address is shown on page 16, in Figure 6. It would have been obvious to one of ordinary skill in the art at the time of invention to modify Bland et al. with using these specific details in the tests contexts as taught by "Network Diagnostics Tools Feature Overview" in order to have all of the information available to an administrator as noted on page 2 under the title "Question".

14. **Claim 23** is rejected under 35 U.S.C. 103(a) as being unpatentable over Bland et al. (US 5,732,218) and "Network Diagnostics Tools Feature Overview" as applied to claim 22 above, and further in view of Day et al. (US 2002/0095487).



Regarding **claim 23**, Bland et al. and "Network Diagnostics Tools Overview" disclose all of the limitations as described above and "Network Diagnostics Tools Overview" further teaches the context can include a server computer system host name (page 17, Figure 7, where the Gateways, DNS, and DHCP addresses are seen as server hosts names). Bland et al. and "Network Diagnostics Tools Overview" do not teach the use of a server network domain name or a server type in the contexts. The general concept of including these configuration details in test context is well known in the art as illustrated by Day et al. Day et al. teaches a diagnostic system where the server's information can be sent for diagnostic purposes (section 0051, lines 5-20, with server name and type being sent). It would have been obvious to one of ordinary skill in the art at the time of invention to modify Bland et al. and "Network Diagnostics Tools Overview" with using server name and type in the context as taught by Day et al. in order to identify easily the server at issue as noted in Day et al.'s disclosure in section 0010.

15. **Claims 24 and 25** are rejected under 35 U.S.C. 103(a) as being unpatentable over Bland et al. (US 5,732,218), "Network Diagnostics Tools Feature Overview", and Day et al. (US 2002/0095487) as applied to claim 23 above, and further in view of Beaven (US 5,627,766).

Regarding **claims 24 and 25**, Bland et al., "Network Diagnostics Tools Feature Overview", and Day et al. teach all of the limitations as described above except for including a globally unique identifier associated with a client computer operating system process and information regarding the client computer operating system process in the

context as required by claim 24 and having the globally unique identifier correspond to a client server communication session. The general concept of using an identifier and information in the performance data context is well known in the art as illustrated by Beaven. Beaven teaches performance monitoring with a global test identifier. This identifier is used to associate tests with certain nodes and figure out which test have already run on a node (column 8, lines 23-30, where the global test identifier is seen as a global identifier used for an operating system process and a client server communications system since it identifies test that are run on a client in a client server environment). The identifier can also be seen as information regarding the client's operating system process since the identifier is used to identify what test has been run on the operating system of the client. It would have been obvious to one of ordinary skill in the art at the time of invention to modify Bland et al., "Network Diagnostics Tools Feature Overview", and Day et al. with using global identifiers in the context as taught by Beaven in order to increase testing efficiency by only running tests if they have not been performed.

16. **Claim 26** is rejected under 35 U.S.C. 103(a) as being unpatentable over Bland et al. (US 5,732,218) in view of Cote et al. (US 5,938,729).

Regarding **claim 26**, Bland et al. discloses all of the limitations as described above except for having the second request include an indication of at least one service desired of a server by a client. The general concept of including this information in the second request is well known in the art as illustrated by Cote et al. Cote et al. teaches that a server can generate a response message from being queried for performance

data. This data includes reports for each service running on the server, and these services are pre-selected by a user to be monitored for performance (column 7, line 65 - column 8, line 14). This is seen as a request that includes service information of services desired by a client, since a client was used to determine what services to report performance data about. It would have been obvious to one of ordinary skill in the art at the time of invention to modify Bland et al. with including service reporting as taught by Cote et al. in order to consolidate the multiple responses from services typically seen in the server monitoring technology as noted in Cote et al.'s disclosure in column 1, lines 20-24.

17. **Claim 33** is rejected under 35 U.S.C. 103(a) as being unpatentable over Bland et al. (US 5,732,218) in view of Caccavale (US 5,664,106).

Regarding **claim 33**, Bland et al. teaches all of the limitations as described above except for receiving the maximum performance data age threshold from the server. The general concept of retrieving thresholds from a server in a computer monitoring system is well known in the art as illustrated by Caccavale. Caccavale teaches that thresholds for monitoring a server can be set at the server in response to network conditions (column 2, lines 36-44, where the thresholds are retrieved from the server in order to tune the server). It would have been obvious to one of ordinary skill in the art at the time of invention to modify Bland et al. with using server generated or retrieved thresholds as taught by Caccavale in order to make the server more efficient by tuning the server to thresholds set at the server as noted in Caccavale's disclosure in column 1, lines 41-43.

18. **Claims 34 and 40** are rejected under 35 U.S.C. 103(a) as being unpatentable over Bland et al. (US 5,732,218) in view of Balasubramanian et al. (US 6,874,099).

Regarding **claim 34**, Bland et al. teaches:

A computerized client system, comprising: a performance data store (column 5, line 66- column 6, line 4, where the report is seen as the performance data store);  
a performance data measurement module configured to, at least:  
generate performance data concerning requests from the client system and  
corresponding responses to the client system (column 5, lines 41-66, where the client transfers performance data to the server which corresponds to the client/server communication session); and  
store the generated performance data in the performance data store (column 5, line 62- column 6, line 4, where the data is sent to the server for report generation).

Bland et al. does not teach:

a server performance data preference store; a performance data stream parse module configured to, at least, parse an incoming data stream for server performance data preferences and store them in the server performance data preference store;  
a performance data stream format module configured to, at least, format and insert performance data from the performance data store into an outgoing data stream in accordance with server performance data preferences.

The general concept of using preferences in storing data in a performance store is well known in the art as illustrated by Balasubramanian et al. Balasubramanian et al.

Art Unit: 2109

teaches that preference values are stored in a database (column 9, lines 50-51, where performance values are seen as preferences since they control the testing of the server according to user input as seen in column 9, lines 44-46). Balasubramanian et al. also teaches that the user can input property values (column 9, lines 44-46, and this is seen as parsing an incoming data stream for property values and storing them in the preference store). Balasubramanian et al. further teaches that the property values control test length and sleep time (column 9, lines 37-44, where these property values must control specific tests and therefore they must be outputted from the performance data store at some point, seen as formatting the performance data store with preferences from the preference store). It would have been obvious to one of ordinary skill in the art at the time of invention to modify Bland et al. with using a preference store in conjunction with the performance data as taught Balasubramanian et al. in order to facilitate changes to the monitoring programs when devices are modified as noted in Balasubramanian et al.'s disclosure in column 2, lines 38-40.

Regarding **claim 40**, Bland et al. teaches all of the limitations as described above except for having a server performance data preference store and a performance data stream format module configured to, at least, format and insert performance data from the performance data store into an outgoing data stream in accordance with server performance data preferences. The general concept of using preferences in storing data in a performance store is well known in the art as illustrated by Balasubramanian et al. Balasubramanian et al. teaches that preference values are stored in a database

(column 9, lines 50-51, where performance values are seen as preferences since they control the testing of the server according to user input as seen in column 9, lines 44-46). Balasubramanian et al. further teaches that the property values control test length and sleep time (column 9, lines 37-44, where these property values must control specific tests and therefore they must be outputted from the performance data store at some point, seen as formatting the performance data store with preferences from the preference store). It would have been obvious to one of ordinary skill in the art at the time of invention to modify Bland et al. with using a preference store in conjunction with the performance data as taught Balasubramanian et al. in order to facilitate changes to the monitoring programs when devices are modified as noted in Balasubramanian et al.'s disclosure in column 2, lines 38-40.

19. **Claims 35-36** are rejected under 35 U.S.C. 103(a) as being unpatentable over Bland et al. (US 5,732,218) in view of Balasubramanian et al. (US 6,874,099) as applied to claim 34 above, and further in view of Beaven (US 5,627,766).

Regarding **claim 35**, Bland et al. and Balasubramanian et al. teach all of the limitations as described above except for having a performance data context map capable of maintaining, for each performance data context, a performance data context identifier to performance data context association;  
a performance data contextualize module configured to, at least:  
instantiate at least one performance data context;  
and

for each performance data context, make a performance data context identifier to performance data context association entry in the performance data context map.

The general concept of using a performance data context identifier in a monitoring system is well known in the art as illustrated by Beaven. Beaven teaches that a global identifier is maintained for each test on each node in which the node has participated (column 8, lines 23-30). This is seen as a performance data context map since it maps test to participating nodes, and it also is seen as a performance data context identifier since it identifies which test has been run on a node. Beaven teaches that the tests are performed (column 8, lines 44-48), and that the tests create global test identifiers (column 8, lines 23-30). This is seen as instantiating a performance data context, or running a test on a node, and creating the identifier to context entry in the map. It would have been obvious to one of ordinary skill in the art at the time of invention to modify Bland et al. and Balasubramanian et al. with using context identifiers as taught by Beaven in order to increase testing efficiency by only running tests if they have not been performed.

Regarding **claim 36**, Bland et al., Balasubramanian et al., and Beaven teach all of the limitations as described above and Beaven further teaches having a performance data stream format module configure to, at least:

Format and insert performance data context independently from other performance data into an outgoing data stream; and

Replace performance data context with its associated performance data context identifier, as specified in the performance data context map.

The general concept of using a performance data context identifier in a monitoring system is well known in the art as illustrated by Beaven. Beaven teaches that a global identifier is maintained for each test on each node in which the node has participated (column 8, lines 23-30). This is seen as a performance data context map since it maps test to participating nodes, and it also is seen as a performance data context identifier since it identifies which test has been run on a node. Beaven teaches that the tests are performed (column 8, lines 44-48), and that the tests create global test identifiers (column 8, lines 23-30). This is seen as independently sorting the performance data context in any communication stream since the identifiers uniquely identify the node the test has run on, seen as the performance data context. It is also seen here that the context identifier, seen as the global identifier is created from the context, or the node it is run on, and this is seen as replacing the context with a global identifier. It would have been obvious to one of ordinary skill in the art at the time of invention to modify Bland et al., Balasubramanian et al., and Beaven with using context identifiers as further taught by Beaven in order to increase testing efficiency by only running tests if they have not been performed.

20. **Claim 39** is rejected under 35 U.S.C. 103(a) as being unpatentable over Bland et al. (US 5,732,218) in view of Beaven (US 5,627,766).



Regarding **claim 39**, Bland et al. discloses all of the limitations as described above including wherein the client-generated performance data comprises performance data context (column 4, lines 54-59, where client demographics are seen as part of the performance data context since it identifies the test run and information about the clients), the performance data stream parse module is further configured to, at least, parse performance data context from the incoming data stream (column 6, lines 2-4, where generating reports is seen as parsing the data, which includes context, and the reports come from data sent to the server from the clients, seen as an incoming data stream).

Bland et al. does not teach a performance data context map capable of maintaining, for each parsed performance data context, a performance data context identifier to performance data context association; and a performance data contextualize module configured to, at least, for each parsed performance data context, make a performance data context identifier to performance data context association entry in the performance data context map.

The general concept of using a performance data context identifier in a monitoring system is well known in the art as illustrated by Beaven. Beaven teaches that a global identifier is maintained for each test on each node in which the node has participated (column 8, lines 23-30). This is seen as a performance data context map since it maps test to participating nodes, and it also is seen as a performance data context identifier since it identifies which test has been run on a node. Beaven teaches that the tests are performed (column 8, lines 44-48), and that the tests create global test

identifiers (column 8, lines 23-30). This is seen as instantiating a performance data context, or running a test on a node, and creating the identifier to context entry in the map. It would have been obvious to one of ordinary skill in the art at the time of invention to modify Bland et al. and Balasubramanian et al. with using context identifiers as taught by Beaven in order to increase testing efficiency by only running tests if they have not been performed.

21. **Claim 42** is rejected under 35 U.S.C. 103(a) as being unpatentable over Kaluskar et al. (US 7,100,171) in view of "Internet Header Format".

Regarding **claim 42**, Kaluskar et al. teaches that an RPC can have extension fields (column 2, lines 8-10). Each field can have a corresponding version field and this can be placed in the header as well (column 2, lines 11-12). These version fields in the RPC tag can correspond to any data value field and can be in the header as well. These are seen as version fields and data blocks. Kaluskar et al. does not teach using format fields or size fields. The general concept of including size and format in a message is well known in the art as illustrated by "Internet Header Format". "Internet Header Format" explains that a standard Internet packet includes a protocol portion (page 4, where this is seen as a format field) and a size field (page 2 with Total Length, and this is seen as a size field). It would have been obvious to one of ordinary skill in the art at the time of invention to modify Kaluskar et al. with using other fields such as format and size as taught by "Internet Header Format" in order to include all relevant information with the message as to increase efficiency of message transmission.

22. **Claim 43** is rejected under 35 U.S.C. 103(a) as being unpatentable over Kaluskar et al. (US 7,100,171) in view of "Internet Header Format" as applied to claim 42 above, and further in view of "Protecting Java Code Via Code Obfuscation" (Low).

Regarding **claim 43**, Kaluskar et al. and "Internet Header Format" teach all of the limitations as described above except for having the format flag be selected from a group consisting of compress data and obfuscate data. The general concept of using compressed or obfuscated data is well known in the art as illustrated by Low. Low teaches that obfuscation of any code can be performed (page 1, paragraph 7, lines 1-3) and compression of data can be performed as well (page 1, paragraph 5, lines 1-2, where encryption is seen as compression). It would have been obvious to one of ordinary skill in the art at the time of invention to modify Kaluskar et al. and "Internet Header Format" with using compressed or obfuscated data as taught by Low in order to better protect the data as noted in Low in page 1, paragraph 3.

23. **Claim 44** is rejected under 35 U.S.C. 103(a) as being unpatentable over Kaluskar et al. (US 7,100,171) in view of "Internet Header Format" as applied to claim 42 above, and further in view of "TCP Header Format".

Regarding **claim 44**, Kaluskar et al. and "Internet Header Format" teach all of the limitations as described above except for having a variable byte storage area and a string type field comprising an offset to a location in the variable byte storage area where the value of the string type field is stored. The general concept of having a

Art Unit: 2109

variable byte storage area and an offset to data is well known in the art as illustrated by "TCP Header Format". "TCP Header Format" teaches that the entire packet except the padding is a variable byte storage area since it requires padding to make the entire packet a certain byte length (page 5). "TCP Header Format" also teaches that the offset field can be used to specify where the actual data it points to is located (page 2). It would have been obvious to one of ordinary skill in the art at the time of invention to modify Kaluskar et al. and "Internet Header Format" with using a variable byte area and an offset field as taught by "TCP Header Format" in order to increase compatibility with standard message protocols.

24. **Claim 45** is rejected under 35 U.S.C. 103(a) as being unpatentable over Kaluskar et al. (US 7,100,171) in view of "Internet Header Format" as applied to claim 42 above, and further in view of Beaven (US 5,627,766) and of "Network Diagnostics Tools Feature Overview".

Regarding **claim 45**, Kaluskar et al. and "Internet Header Format" disclose all of the limitations as described above except for having the block body include an indication that it includes client information performance data context pertaining to a client and having the block body comprise a context identifier, a computer name, a user name, network adaptor, adaptor speed, and protocol address. The general concept of including an indicator that the message includes client information context is well known in the art as illustrated by Beaven. Beaven teaches that a globally unique identifier can be included in messages passed between nodes and these identifiers identify what

Art Unit: 2109

tests the client has already participated in, seen as part of the performance data context (column 8, lines 24-30). It would have been obvious to one of ordinary skill in the art at the time of invention to modify Kaluskar et al. and "Internet Header Format" with using indications of contexts as taught by Beaven in order to increase testing efficiency by only running tests if they have not been performed. The general concept of including other configuration details with the contexts is well known in the art as illustrated by "Network Diagnostics Tools Feature Overview". "Network Diagnostics Tools Feature Overview" teaches that certain configuration details can be included in tests run to monitor a network. The user name is shown on page 10, under Computer Information as "CSName", and this is also interpreted as a computer name. The network adaptor name and speed is shown on page 20 in Figure 9. The network protocol address is shown on page 16, in Figure 6. It would have been obvious to one of ordinary skill in the art at the time of invention to modify Kaluskar et al., "Internet Header Format", and Beaven with using these specific details in the tests contexts as taught by "Network Diagnostics Tools Feature Overview" in order to have all of the information available to an administrator as noted on page 2 under the title "Question".

25. **Claim 46** is rejected under 35 U.S.C. 103(a) as being unpatentable over Kaluskar et al. (US 7,100,171) in view of "Internet Header Format" as applied to claim 42 above, and further in view of Day et al. (Us 2002/0095487).

Regarding **claim 46**, Kaluskar et al. and "Internet Header Format" disclose all of the limitations as described above except the use of server information in the data block

including server name, a server network domain name or a server type in the contexts. The general concept of including these configuration details in test context is well known in the art as illustrated by Day et al. Day et al. teaches a diagnostic system where the server's information can be sent for diagnostic purposes (section 0051, lines 5-20, with server hostname, domain name, and type being sent). Including these details would in itself be an indication of server information in the message. It would have been obvious to one of ordinary skill in the art at the time of invention to modify Kaluskar et al. and "Internet Header Format" with using server name and type in the context as taught by Day et al. in order to identify easily the server at issue as noted in Day et al.'s disclosure in section 0010.

26. **Claim 47-48, and 50** are rejected under 35 U.S.C. 103(a) as being unpatentable over Kaluskar et al. (US 7,100,171) in view of "Internet Header Format" as applied to claim 42 above, and further in view of Bland et al. (US 5,732,218).

Regarding **claim 47**, Kaluskar et al. and "Internet Header Format" disclose all of the limitations as described above except including an indication of the data block body contains client/server request/response information including a request identifier, a round trip latency, and a server processing time. The general concept of including this type of data in a message is well known in the art as illustrated by Bland et al. Bland et al. teaches a computer monitoring system in which messages are passed between the client and the server and they can include identifiers (column 4, lines 54-59, with client demographics seen as an identifier of a client), latency information (column 4, lines 11-

Art Unit: 2109

20, with round trip time computed), and server processing time (column 3, lines 46-49). This information is compiled and sent to a service manager (column 6, lines 1-4). The placement of this information in a message in itself is an indication that the message includes this information. It would have been obvious to one of ordinary skill in the art at the time of invention to modify Kaluskar et al. and "Internet Header Format" with using client/server performance data as taught by Bland et al. in order to give the service manager better results by monitoring servers by a client as noted in Bland et al.'s disclosure in column 2, lines 37-40.

Regarding **claim 48**, Kaluskar et al., "Internet Header Format", and Bland et al. teach all of the limitations as described above except for using a data block size of 14 bytes. It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Kaluskar et al., "Internet Header Format", and Bland et al. to include the use of a block size of 14 bytes in their systems, as choosing a message size is a common and everyday occurrence throughout the message and protocol art and the specific use of 14 bytes would have been an obvious matter of design preference depending upon such factors as speed, protocol, or memory; the ordinarily skilled artisan would choose the best size which would most optimize the cost and performance of the device for a particular application at hand, based upon the above noted common design criteria.

Regarding **claim 50**, Kaluskar et al. and "Internet Header Format" teach all of the limitations as described above except for wherein the performance data block type field indicates that the performance data block body contains server performance data preference information for a server in a client server network, and wherein the performance data block body comprises: an indication of whether to send client-generated performance data to the server pertaining to client communications with the server; an indication of whether to send client-generated performance data to the server pertaining to client communications with other servers in the client server network; and a performance data age threshold beyond which client-stored performance data should not be sent to the server. The general concept of including indications of whether to send client data about the server or other servers in a message and also a data age threshold is well known in the art as illustrated by Bland et al. Bland et al. teaches that the server can request to start the gathering of information by a client by messaging the client, and also the information retrieved can be from other servers (column 5, lines 4-16). Bland et al. also teaches that a data age threshold may be implemented (column 5, lines 16-19, where every 30 minutes data is transferred when data is in the file, which is seen as a data age threshold in which data is supposed to be transferred at those specific times). The placement of this information in a message in itself is an indication that the message includes this information. It would have been obvious to one of ordinary skill in the art at the time of invention to modify Kaluskar et al. and "Internet Header Format" with using client/server performance data request indications as taught



by Bland et al. in order to give the service manager better results by monitoring servers by a client as noted in Bland et al.'s disclosure in column 2, lines 37-40.

27. **Claim 49** is rejected under 35 U.S.C. 103(a) as being unpatentable over Kaluskar et al. (US 7,100,171) in view of "Internet Header Format" as applied to claim 42 above, and further in view of Cote et al. (US 5,870,556).

Regarding **claim 49**, Kaluskar et al. and "Internet Header Format" disclose all of the limitations as described above except for wherein the performance data block type field indicates that the performance data block body contains client-generated performance data pertaining to a failed request/response pair between a client and a server in a client server network, and wherein the performance data block body comprises: a client-generated request identifier associated with the failed request/response pair; a time to fail corresponding to a difference between a time at which the client initiated the failed request/response pair and a time at which the client determined that the request had failed; and a failure code corresponding to a reason for the failure of the request. The general concept of including failure information in an RPC tag is well known in the art as illustrated by Cote et al. Cote et al. teaches that a monitoring system can report failures. The messages use identification strings to identify the messages passed (column 5, lines 6-16), and a report can be generated from message failures. This is seen as having a message include a time to fail, since the message will have a time stamp and also be passed an amount of time allotted for a successful response (column 6, lines 20-31). It also includes in a report reasons for failure (column 6, lines 28-31, where reasons for failure are given, seen as the failure

Art Unit: 2109

code). Any inclusion of this information is in itself an indication that a message includes this type of information. It would have been obvious to one of ordinary skill in the art at the time of invention to modify Kaluskar et al. and "Internet Header Format" with using failure codes and times as taught by Cote et al. in order to better diagnose failures as noted in Cote et al.'s disclosure on column 1, lines 18-35.

### ***Conclusion***

28. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Frank et al. (US 2003/0158942) uses RPC messaging to reconfigure a server in a computer network.


Any inquiry concerning this communication or earlier communications from the examiner should be directed to Adam S. Weintrop whose telephone number is 571-270-1604. The examiner can normally be reached on Monday through Friday 7:30am- 5:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Frantz Jules can be reached on 571-272-6681. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Art Unit: 2109

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

AW 5/22/07



JASON CARDONE  
SUPERVISORY PATENT EXAMINER